ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

APLICACIÓN WEB PARA ENSEÑANZA/APRENDIZAJE DE CÁLCULO Y ÁLGEBRA

A WEB-BASED ENVIRONMENT FOR CALCULUS AND ALGEBRA ASSESSMENT AND LEARNING

Realizado por
ADRIÁN ALMIRÓN GARCÍA
Tutorizado por
EDUARDO GUZMÁN DE LOS RISCOS
Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE 2015

Fecha defensa:
El Secretario del Tribunal

Resumen:

El principal objetivo de este trabajo fin de grado es la implementación de una aplicación web que permita realizar actividades de enseñanza/aprendizaje en las ramas de cálculo y álgebra de las matemáticas, enfocada principalmente en los contenidos impartidos en las asignaturas de matemáticas en Bachillerato, concretamente de la unidad de realización de derivadas.

Se han desarrollado dos modelos:

Uno para el profesor, que permite la generación de los ejercicios, así como la consulta de las posibles soluciones. Esta aplicación ofrece al docente la posibilidad de mediante una serie de ajustes, generar de forma dinámica las actividades deseadas.

Otro para el alumno, que permite la realización de las actividades y la inclusión de respuestas paso a paso, además de la visualización de las correcciones a sus procedimientos.

Para el desarrollo de estas aplicaciones se ha usado el lenguaje Java con ayuda de JSON para el intercambio de datos. También se ha utilizado el motor de respuestas Wolfram Alpha para realizar las correcciones paso a paso de las respuestas de los alumnos.

Palabras Claves: Aplicación web, enseñanza, aprendizaje, derivadas, automatización.

Abstract:

The main objective in this project is the implementation of a web application to practise assessment and learning activities about algebra and calculus areas- mainly about the contents taught in the mathematics subject in Spanish High School, specifically the derivatives unit.

We have developed two models:

The first one for the teacher, which allows to generate derivative activities, and it also allows to consult the possible solutions to them. This application gives to the teacher the possibility to generate activities automatically; you can do it selecting some options in the application.

The second one for the student, which allows him to include his own answers, step by step, and the final solution to the activity. Also he can consult the revision to the answers.

To develop this application we have used Java language with JSON help to exchange data between both models. Also we have used Wolfram Alpha to correct step by step the students' answers.

Keywords: Web application, assessment, learning, derivatives, automation.

Contenido

Índice de ilustraciones

1 Introducción

En este capítulo se recogen los primeros apartados de la memoria, como son la motivación que ha provocado la realización del presente trabajo, la enumeración de los objetivos que se persiguen con el mismo, la descripción de las fases de trabajo llevadas a cabo y la explicación de la forma en la que está estructurada la memoria.

1.1 Motivación

La motivación del presente trabajo fin de grado surge de varias ideas:

En primer lugar, la línea de trabajos del ámbito de e-learning ofertada por el departamento de Lenguajes y Ciencias de la Computación.

En segundo lugar, de las reuniones previas con otros profesores para la realización de posibles trabajos fin de grado, en las que se tiene conocimiento de un proyecto fin de carrera titulado “Sistema instructor inteligente para el cálculo de derivadas”, realizado por J.F. Lacal Díaz y dirigido por J.L. Pérez De la Cruz Molina.

Finalmente, de la reunión entre el tutor y el autor de este trabajo, en la que tras observar la experiencia del alumno en clases particulares de matemáticas, se acuerda la realización de un anteproyecto. En el mismo se debía idear una aplicación, que permita la generación de ejercicios de cálculo y álgebra de forma dinámica, y la posterior corrección de las soluciones del alumno, de forma automática.

1.2 Objetivos

Los principales objetivos marcados en este proyecto son:

- Crear una aplicación que ayude a la enseñanza y aprendizaje de alguno de los contenidos impartidos en la asignatura de matemáticas de nivel de Bachillerato.
- Generación automática de ejercicios en base a ciertos parámetros.
- Diseñar una interfaz para poder ver las soluciones y los posibles pasos para la solución de los ejercicios.
- Corrección automática de soluciones paso a paso.
- Implementar una interfaz para proporcionar soluciones y comprobar las correcciones.

1.3 Fases de trabajo

La elaboración del presente trabajo fin de grado se ha ejecutado a través de un conjunto secuencial de fases. Con la realización de cada nueva fase, se han ido revisando las anteriores y añadiendo los pasos que se han considerado precisos. El conjunto de fases realizadas se enumera a continuación.

1.3.1 Primera: Estudio del contexto matemático

En primer lugar, y una vez conocidos los objetivos del proyecto, era necesario decidir qué unidad concreta de la rama de las matemáticas se quería cubrir con la implementación del trabajo. Para ello, era preciso elegir una rama que fuese adaptable a la generación dinámica de los ejercicios, así como a su posterior corrección de forma automática a través de Wolfram Alpha (WA). Una vez tomada la decisión de centrarnos en el ámbito de las derivadas, hubo que estudiar la forma de afrontar los propios

apartados de esta rama. A la explicación de los mismos se dedica un capítulo posteriormente.

1.3.2 Segunda: Estudio de las nuevas herramientas a utilizar

Con las ideas ya claras en torno a los conceptos matemáticos necesarios para realizar la aplicación, se estudiaron en profundidad las nuevas herramientas que el alumno no había utilizado con anterioridad.

Principalmente, se trata de la notación JSON, para el intercambio de datos entre ambas partes de la aplicación, así como la biblioteca GSON que facilita el uso de la notación anteriormente mencionada.

Por otro lado, también se estudió el funcionamiento de la API de Wolfram Alpha. Empezando por las respuestas de dicha API, tarea ampliamente facilitada por el explorador interactivo incluido en su propia web, y concluyendo por la forma de realizar las consultas, labor también simplificada gracias a la librería proporcionada en la web del motor de respuestas.

1.3.3 Tercera: Montaje de la estructura de la aplicación

El siguiente paso fue realizar un primer montaje de la estructura de la aplicación. Crear los Servlets, archivos JSP y JSON necesarios para poder empezar a programar las funcionalidades deseadas. Ésta es una de las fases que en primer lugar fue realizada de forma más escueta, y fue incrementándose progresivamente al ir realizando las siguientes.

1.3.4 Cuarta: Creación de las clases Java necesarias

Al igual que la fase anterior, se comenzó creando una pequeña clase “Ejercicio” que simplemente almacenaba las funciones de cada apartado del ejercicio. Posteriormente, al mejorar la comunicación con WA, esta clase se incrementó guardando también las soluciones de los mismos o los posibles pasos para realizar la derivación de dichas funciones. Además, se crearon las clases necesarias para obtener esas soluciones. A continuación, también habrá un apartado completo en el que se explicará el funcionamiento concreto de cada una de estas clases.

1.3.5 Quinta: Generación automática de los ejercicios

La siguiente tarea a realizar fue la implementación del código que hacía que se generasen de forma automática funciones de diferentes tipos y dificultades para su inclusión en los apartados de cada ejercicio. Dicho código se incluye dentro de un Servlet que será explicado en profundidad en el apartado correspondiente.

1.3.6 Sexta: Comunicación con Wolfram Alpha

En siguiente lugar, se realizó la parte del trabajo necesaria para comunicar la aplicación con WA. De esta comunicación se obtuvieron tres resultados: la obtención de las soluciones a los apartados (así como unos posibles pasos para llegar a esas soluciones), la obtención de enlaces a imágenes que muestren cada función de forma matemática tradicional, y la corrección de los ejercicios.

1.3.7 Séptima: Corrección automática de las soluciones

Una vez realizada la fase anterior, ya se disponía de un método que devuelva si dos funciones pasadas como parámetros son equivalentes. Por tanto, para esta fase solo

fue necesaria la corrección, dos a dos, de los pasos proporcionados por el alumno, así como la comprobación de que la solución final era adecuada para el enunciado original del apartado.

1.3.8 Octava: Mejora de la interfaz gráfica de la aplicación

Con las funcionalidades deseadas ya implementadas, la fase final del trabajo consistió en hacer una mejora de la parte visual de la aplicación. Para el desarrollo de dicha tarea se usaron las herramientas proporcionadas por Bootstrap y CSS.

1.4 Estructura de la memoria

En esta memoria se pretende dejar constancia de las tareas llevadas a cabo para la realización del proyecto. La estructura de la misma se divide en los siguientes capítulos:

- Introducción. Es el capítulo en el que se encuadra este apartado, su finalidad es la de establecer una primera aproximación al trabajo realizado durante este proyecto.
- Tecnologías a utilizar. Breve descripción de cada una de las tecnologías utilizadas durante el desarrollo del proyecto.
- Especificación del sistema. Requisitos, casos de uso y diagramas de especificación del sistema.
- Diseño de la aplicación. Descripción del trabajo realizado previamente a la implementación del sistema.
- Implementación. Descripción de cada uno de los archivos de los que se compone la aplicación.
- Conclusiones. Explicación de las dificultades encontradas para el desarrollo de la aplicación, así como conclusiones finales y exposición de posibles líneas de futuro.
- Referencias Bibliográficas. Bibliografía utilizada para la redacción de esta memoria.
- Anexos Técnicos. Compuesto del manual de usuario y el índice de ilustraciones.

2 Tecnologías a utilizar

En este capítulo se pretende realizar una breve explicación de cada una de las herramientas y tecnologías utilizadas para realizar el desarrollo de este trabajo fin de grado.

2.1 Servlets

Servlets es una tecnología de la plataforma Java que mejora y amplía las capacidades de un servidor. Éstos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web.

Los Servlets tienen acceso a toda la familia de APIs Java, incluida la API JDBC para acceder a bases de datos empresariales. Los Servlets, también pueden acceder a una biblioteca específica HTTP de llamadas y recibir todos los beneficios del lenguaje Java, incluyendo la portabilidad, rendimiento, reutilización, etc.

El uso más común de los Servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Los dos métodos principales dentro de un Servlet son:

- doGet() se llama después de una consulta de tipo HTTP GET. Generalmente, esto ocurre cuando un usuario pulsa en un enlace o cuando se escribe directamente en la barra del navegador.
- doPost() se llama después de una consulta de tipo HTTP POST.

Los dos métodos se llaman desde la implementación por defecto del método service() que se encuentra en la clase base HttpServlet.

Normalmente, el navegador pide siempre páginas mediante GET y puede enviar datos bajo las dos formas (GET y POST).

El ciclo de vida de un Servlet es el siguiente:

1. El cliente solicita una petición a un servidor vía URL.
2. El servidor recibe la petición.
   a) Si es la primera, se utiliza el motor de Servlets para cargarlo y se llama al método init().
   b) Si ya está iniciado, cualquier petición se convierte en un nuevo hilo. Un Servlet puede manejar múltiples peticiones de clientes.
3. Se llama al método service() para procesar la petición devolviendo el resultado al cliente.
4. Cuando se apaga el motor de un Servlet se llama al método destroy(), que lo destruye y libera los recursos abiertos.

En la siguiente ilustración podemos observar este ciclo de vida:

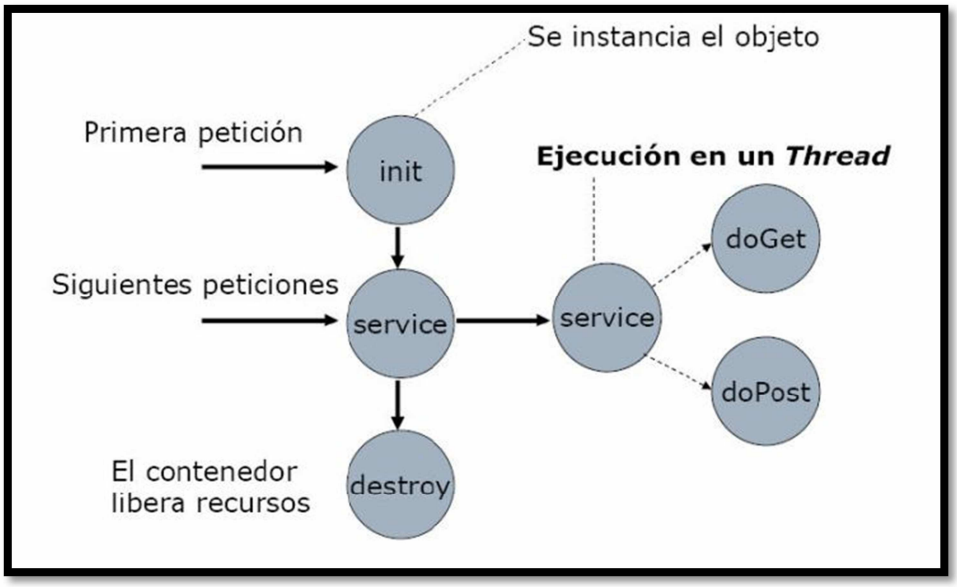

Ilustración 1. Ciclo de vida de un Servlet

2.2 JSP

Java Server Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML y otros tipos de documentos. Permite desarrollar rápidamente y mantener con facilidad páginas Web. Como parte de la familia de tecnologías Java, la tecnología JSP permite el desarrollo rápido de aplicaciones Web de forma independiente a la plataforma utilizada. La tecnología JSP separa la interfaz de usuario de generación de contenidos, permitiendo a los diseñadores cambiar el diseño general de la página sin alterar el contenido dinámico subyacente.

Algunas de las ventajas de usar esta tecnología son:

- Se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él.
- Cada página se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo.
- Su persistencia le permite también hacer una serie de tareas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

Estos beneficios se obtienen de la inclusión de código Java dentro de la propia página JSP, a través de los llamados “Tags JSP”. Los principales son los siguientes:

- Expresión: se utiliza para integrar el valor de la expresión Java dentro del código, por ejemplo el valor de una variable recibida.

< % = expresión Java % >

- Código: utilizado para ejecutar código Java.

< % código Java % >

- Directiva: son indicaciones que se le dan al motor de Servlets acerca de los parámetros generales de ejecución para esa página.

< % @page atributo = “valor” % >

- Acciones: Permiten incluir una página HTML o JSP como parte de la página actual.

< % jsp : include page = “cabecera.html” % >

2.3 Tomcat

Aunque la parte inicial del proyecto fue desarrollada usando un servidor de aplicaciones Web Glassfish, debido a la aparición de una serie de problemas a la hora de desplegar la aplicación, se decidió cambiar a un servidor Tomcat.

Apache Tomcat es un contenedor web con soporte de Servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en Servlets.

Tomcat puede funcionar como servidor web por sí mismo. Puede ser usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Al estar escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

2.4 JSON

JavaScript Object Notation (JSON), es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. La simplicidad de su uso ha dado lugar a la generalización del mismo. Esta simplicidad es más destacada a la hora de hacer análisis de códigos JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por muchos desarrolladores.

JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente.

Cada vez, hay más soporte de JSON mediante el uso de paquetes escritos por terceras partes. La lista de lenguajes soportados incluye: ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro.

2.5 GSON

GSON es una biblioteca Java que puede ser usada para convertir objetos Java en su representación JSON. También, se utiliza para convertir una cadena JSON en su equivalente objeto Java. Es un proyecto de código abierto. Sus principales ventajas son las siguientes:

- Permite la conversión entre objetos Java y JSON de una manera sencilla, simplemente invocando los métodos toJson() o fromJson().
- Permite la conversión de objetos inmutables ya existentes.
- Soporte para tipos genéricos de Java.
- Permite la representación personalizada de objetos.
- Soporte para "Objetos arbitrariamente complejos".

2.6 Wolfram Alpha

Wolfram Alpha es un buscador de respuestas desarrollado por la compañía Wolfram Research. Es un servicio en línea que responde a las preguntas directamente, mediante el procesamiento de la respuesta extraída de una base de datos estructurados, en lugar de proporcionar una lista de los documentos o páginas web que podrían contener la respuesta, tal y como lo hacen otros buscadores. Fue anunciado en marzo de 2009 por el físico británico Stephen Wolfram y está en funcionamiento desde el 15 de mayo de 2009.

En concreto, en este trabajo se usa la API de este motor de respuestas. Éste proporciona una API que proporciona las capacidades computacionales y de representación de Wolfram Alpha y que puede ser integrada en aplicaciones web, móviles, de escritorio, y de la empresa.

La API permite a los clientes enviar consultas de forma libre similares a las consultas que se podrían introducir en su web, y capturar los resultados que son devueltos en una gran variedad de formatos. El API se implementa en un protocolo REST estándar utilizando peticiones HTTP GET. Cada resultado se devuelve como una estructura XML descriptiva con el contenido en el formato requerido.

Esta plataforma puede ser accedida en múltiples niveles, desde resultados individuales a páginas completas de Wolfram Alpha. Esta tarea puede ser realizada de forma más sencilla gracias a los diversas librerías proporcionadas para los principales lenguajes y plataformas.

2.7 CSS

Cascading Style Sheets (CSS) es un lenguaje usado para definir cómo se muestran los elementos de HTML. A partir de la versión 4.0 de HTML se comienza a usar el atributo “Style”, como método para solventar el problema de los estilos de los documentos.

El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

Hay tres formas de usar hojas de estilos:

- Hoja de estilos externa. Se trata de hacer una definición del estilo de la página en un documento CSS externo. Posteriormente se enlaza dicho documento a la página HTML requerida a través del elemento “<link>” en su cabecera.
- Hoja de estilos interna. En este caso la definición del estilo se hace dentro de la propia página HTML. Se colocaría dentro del elemento “<style>” de su cabecera.
- Estilo en línea. Consiste en aplicar un estilo a cada uno de los elementos de la página HTML. Para ello se usa el atributo “style” en la etiqueta correspondiente.

2.8 Bootstrap

Bootstrap es un marco de trabajo (framework) o conjunto de herramientas de software libre para diseño de páginas y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales. También permite crear fácilmente diseños de tipo “Responsive Web Design”.

Bootstrap fue desarrollado por Mark Otto y Jacbod Thornton de Twitter, como un framework para fomentar la consistencia a través de herramientas internas. En agosto del 2011, Twitter liberó a Bootstrap como código abierto. En febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub.

Sus ventajas son las siguientes:

- Fácil de usar. Cualquier persona con un conocimiento básico de HTML y CSS puede empezar a usar Bootstrap.
- Características de tipo “Responsive”. Las páginas diseñadas con Bootstrap se adaptan a teléfonos, tablets y ordenadores.
- Compatible con navegadores. Bootstrap es compatible con todos los navegadores actuales (Chrome, Firefox, Internet Explorer, Safari y Opera).

3 Especificación del sistema

En este capítulo se describen los requisitos funcionales y no funcionales del sistema, así como los casos de uso y diagramas más relevantes que se obtuvieron durante la fase de análisis de la aplicación.

3.1 Requisitos funcionales

La aplicación deberá ofrecer a los usuarios las siguientes funcionalidades:

RF01. Acceder de forma distinguida al sistema - El sistema debe ser capaz de permitir el acceso de los usuarios a la aplicación, distinguiendo si se trata de un alumno o un profesor.

RF02. Generar ejercicios - El sistema debe ser capaz de generar ejercicios de derivadas con respecto a los parámetros introducidos.

RF03. Mostrar soluciones - El sistema debe ser capaz de mostrar las soluciones a cada apartado.

RF04. Mostrar pasos - El sistema debe ser capaz de mostrar los posibles pasos para llegar a la solución de un apartado.

RF05. Mostrar formas alternativas - El sistema debe ser capaz de mostrar las posibles formas alternativas a una solución.

RF06. Mostrar ejercicio - El sistema debe ser capaz de mostrar todos los apartados de un ejercicio.

RF07. Corregir solución - El sistema debe ser capaz de corregir las soluciones propuestas por el alumnado.

RF08. Volver a inicio - El sistema debe proporcionar algún mecanismo para volver al inicio de la aplicación.

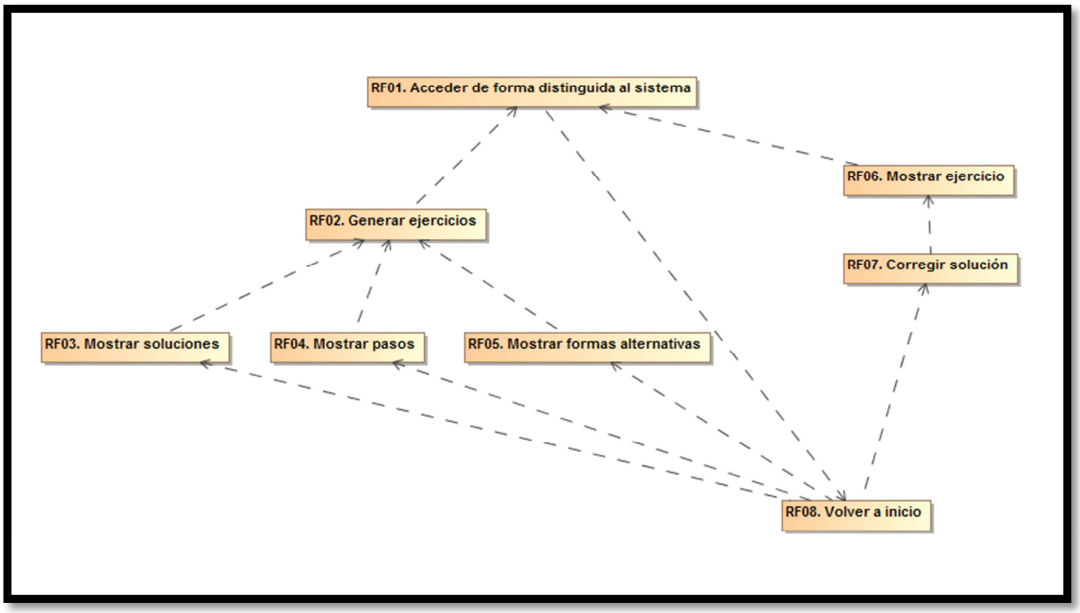

Ilustración 2. Diagrama de requisitos funcionales

3.2 Requisitos no funcionales

Entre los requisitos no relacionados con la funcionalidad del sistema se destacan los siguientes:

RNF01. Usabilidad - El sistema ofrecerá una interfaz gráfica intuitiva y amigable que permita a los usuarios un fácil manejo y un aprendizaje de utilización rápido.

RNF02. Robustez - El sistema debe estar preparado para responder y controlar las entradas incorrectas de los usuarios.

RNF03. Extensibilidad - El sistema debe ser fácilmente extensible. Se le debe poder añadir nuevas funcionalidades de una manera relativamente fácil o mejorar las ya existentes sin afectar a su funcionamiento.

3.3 Actores del sistema

Dentro del sistema se distinguen dos tipos de actores:

- Profesor: Podrá generar ejercicios, así como ver las soluciones a los mismos.
- Alumno: Podrá consultar los ejercicios propuestos, así como proponer sus propias soluciones y obtener las correcciones a las mismas.

3.4 Casos de uso del sistema

A partir de los requisitos funcionales se extraen los siguientes casos de uso. Para cada uno de estos casos de uso se va a dar una descripción con el siguiente formato:

Identificador:	Identificador del caso de uso
Título:	Título del caso de uso
Descripción:	Describe brevemente el caso de uso
Prerrequisito:	Indica si el caso de uso tiene algún prerrequisito
Escenario principal:	
Pasos que describen el funcionamiento normal del caso de uso	
Escenario alternativo:	
Posibles alternativas al funcionamiento normal del caso de uso	

En la siguiente ilustración podemos ver el diagrama de casos de uso:

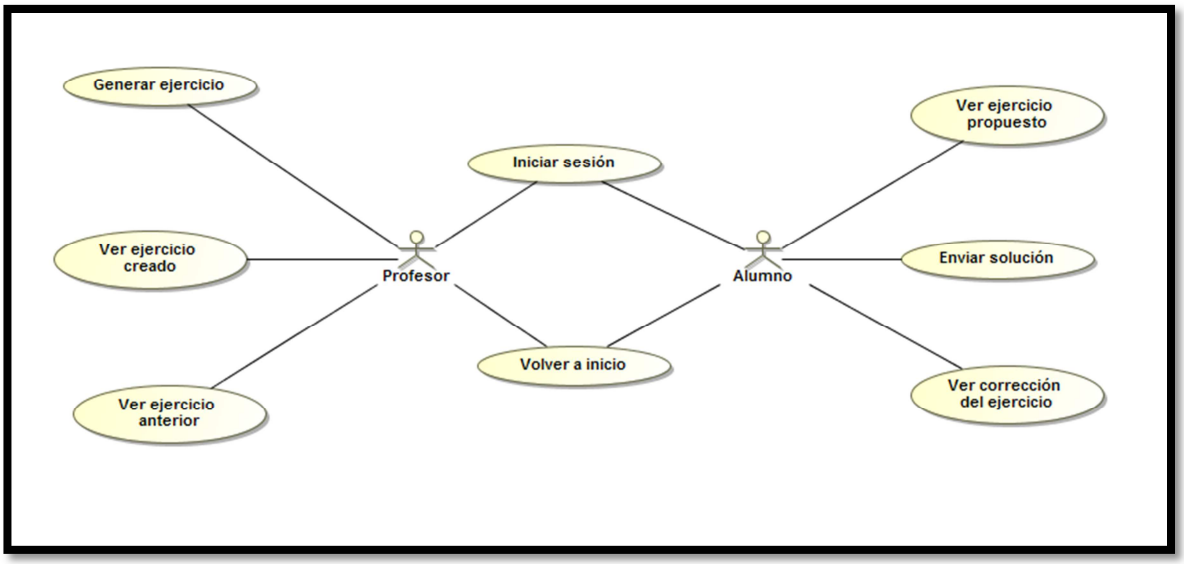

Ilustración 3. Diagrama de casos de uso

Identificador:	CU01
Título:	Iniciar sesión
Descripción:	Permite al usuario acceder a su parte de la aplicación
Escenario principal:	
1. El sistema muestra la página de entrada. 2. El usuario selecciona su tipo de perfil. 3. El usuario pulsa el botón “Aceptar”. 4. El sistema carga la información necesaria. 5. El sistema muestra una página en función del perfil seleccionado.	

Identificador:	CU02
Título:	Volver a inicio
Descripción:	Permite al usuario volver a la página de entrada
Prerrequisito:	Se ha ejecutado el CU01
Escenario principal:	
1. El usuario pulsa el botón “Inicio” en cualquiera de las páginas en las que aparece. 2. El sistema carga la página de entrada.	

Identificador:	CU03
Título:	Generar ejercicio
Descripción:	Permite a un usuario con perfil “Profesor” generar un nuevo ejercicio
Prerrequisito:	Se ha ejecutado el CU01
Escenario principal:	
1. El “Profesor” selecciona los parámetros necesarios para la generación del ejercicio. 2. El “Profesor” pulsa el botón “Generar ejercicio”. 3. El sistema crea un ejercicio en base a los parámetros seleccionados.	

Identificador:	CU04
Título:	Ver ejercicio creado
Descripción:	Permite al usuario con perfil “Profesor” ver los datos del ejercicio que acaba de crear
Prerrequisitos:	Se han ejecutado los casos de uso CU01 y CU03
Escenario principal:	
1. El sistema acaba de crear un ejercicio. 2. El sistema muestra la solución, los posibles pasos y las formas alternativas del primer apartado del ejercicio que se acaba de crear.	
Escenario alternativo:	
2.1 El apartado creado no tiene posibles pasos o formas alternativas y el sistema sólo muestra el enunciado y la solución al mismo.	

Identificador:	CU05
Título:	Ver ejercicio anterior
Descripción:	Permite ver a un usuario con perfil “Profesor” un ejercicio creado anteriormente.
Prerrequisitos:	Se ha ejecutado el CU01 y hay ya un ejercicio creado en el sistema.
Escenario principal:	
1. El “Profesor” pulsa el botón “Ver ejercicio anterior”. 2. El sistema muestra la solución, los posibles pasos y las formas alternativas del primer apartado del ejercicio que ya se encuentra en el sistema.	
Escenario alternativo:	
2.1 El apartado mostrado no tiene posibles pasos o formas alternativas y el sistema sólo muestra el enunciado y la solución al mismo.	

Identificador:	CU06
Título:	Ver ejercicio propuesto
Descripción:	Permite ver a un usuario con el perfil “Alumno” todos los apartados de un ejercicio propuesto.
Prerrequisitos:	Se ha ejecutado el CU01 y hay ya un ejercicio creado en el sistema.
Escenario principal:	
1. El sistema carga los datos del ejercicio almacenado en el mismo. 2. El sistema muestra todos los apartados del ejercicio creado.	

Identificador:	CU07
Título:	Enviar solución
Descripción:	Permite enviar una propuesta de solución a un usuario con perfil “Alumno”.
Prerrequisitos:	Se han ejecutado los casos de uso CU01 y CU06
Escenario principal:	
1. El “Alumno” pulsa el botón “Enviar Solución”. 2. El sistema carga la página con los apartados necesarios para que el alumno proponga los pasos y la solución al apartado seleccionado.	

Identificador:	CU08
Título:	Ver corrección del ejercicio
Descripción:	Permite a un usuario con el perfil “Alumno” ver la corrección a la solución propuesta con anterioridad.
Prerrequisitos:	Se han ejecutado los casos de uso CU01, CU06 y CU07
Escenario principal:	
1. El “Alumno” rellena los campos necesarios para aportar su solución. 2. El “Alumno” presiona el botón “Corregir”. 3. El sistema corrige los pasos y la solución propuestos por el “Alumno”. 4. El sistema muestra si los pasos y la solución son correctos o incorrectos.	

3.5 Matriz de trazabilidad

Con objeto de verificar que todos los requisitos funcionales del sistema se encuentran cubiertos por algún caso de uso, se presenta la siguiente matriz de casos de usos por requisitos:

	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08
CU01	**X**							
CU02								**X**
CU03		**X**						
CU04			**X**	**X**	**X**			
CU05			**X**	**X**	**X**			
CU06						**X**		
CU07							**X**	
CU08							**X**	

Como se puede apreciar, no queda ningún requisito sin cubrir y todo caso de uso surge de algún requisito.

4 Diseño de la aplicación

Este capítulo se dedica a la descripción del trabajo previo realizado antes de la implementación de la aplicación. Dicho trabajo consiste principalmente en dos tareas: realizar un estudio del contexto matemático en el que encuadrar la aplicación y diseñar una estructura de archivos que permita el desarrollo del trabajo.

4.1 Contexto matemático

Tal y como se ha comentado con anterioridad, el ámbito de la aplicación es el de las ramas de cálculo y álgebra con un nivel de Bachillerato. Se hace necesario, por tanto, un estudio de los apartados concretos en los que enmarcar el trabajo.

Tras el estudio del temario impartido en los niveles en los que se pretende encuadrar la aplicación (1º y 2º de Bachillerato), se decide crear un sistema que permita generar ejercicios de derivadas en orden creciente de dificultad. Para dicha tarea, se ha realizado un estudio y planteamiento, con alumnos de este nivel, en los que se ha comprobado que la presentación del contenido, tal y como está explicada en los siguientes apartados, ayuda a la compresión de los mismos.

4.1.1 Derivadas

Se pretende facilitar el estudio de los alumnos, por tanto, es conveniente que los tipos de ejercicios que se planteen empiecen desde un nivel relativamente bajo que se pueda ir aumentando progresivamente.

4.1.2 Inmediatas

En primer lugar, comenzaremos poniendo ejercicios de derivadas inmediatas, introduciendo pequeñas modificaciones que no incrementen su dificultad, con el fin de que el alumno se adapte y comience a aprender las derivadas inmediatas más comunes. En la ilustración 4 podemos observar las mismas.

$y = k$	$y' = 0$
$y = x$	$y' = 1$
$y = x^n$	$y' = nx^{n-1}$
$y = \sqrt{x}$	$y' = \frac{1}{2\sqrt{x}}$
$y = \frac{1}{x}$	$y' = \frac{-1}{x^2}$
$y = e^x$	$y' = e^x$
$y = a^x$	$y' = a^x \cdot \ln(a)$
$y = \ln(x)$	$y' = \frac{1}{x}$
$y = \log_a(x)$	$y' = \frac{1}{\ln(a)} \frac{1}{x}$
$y = \text{sen}(x)$	$y' = \cos(x)$
$y = \cos(x)$	$y' = -\text{sen}(x)$
$y = \text{tg}(x)$	$y' = 1 + \text{tg}^2(x) = \frac{1}{\cos^2(x)} = \sec^2(x)$
$y = \text{arc sen}(x)$	$y' = \frac{1}{\sqrt{1-x^2}}$
$y = \text{arc cos}(x)$	$y' = \frac{-1}{\sqrt{1-x^2}}$
$y = \text{arc tg}(x)$	$y' = \frac{1}{1+x^2}$

Ilustración 4. Tabla de derivadas inmediatas

4.1.3 Suma y resta

El siguiente nivel de dificultad consistirá en sumas y restas de derivadas inmediatas, en las que una vez que el alumno domine el apartado anterior simplemente tendrá que sumar o restar la derivada inmediata de cada una de las funciones individuales. Podemos ver algunos ejemplos en la siguiente ilustración:

$9^x + \tan(x) - \sqrt[4]{x}$	$\cos(x) - 3 + 5x^5$
$\sin(x) - \cos(x) + e^x$	$e^x + 9x^8 - \sqrt[3]{x}$
$1x^2 - \log(x) + \sin(x)$	$5x^2 + \log(x) - \tan(x)$

Ilustración 5. Suma y resta de funciones elementales

4.1.4 Multiplicación y división

A continuación, cuando el alumno ya conozca las reglas de derivación del producto y la división, se le propondrán actividades en las que tenga que derivar multiplicaciones y divisiones de funciones elementales, o que sean la suma o resta de derivadas inmediatas. Dentro de este nivel se puede incluir un subnivel con un poco más de

dificultad en el que haya funciones con varias multiplicaciones y divisiones en forma anidada. En la siguiente ilustración podemos ver dos ejemplos de cada uno de los formatos explicados anteriormente:

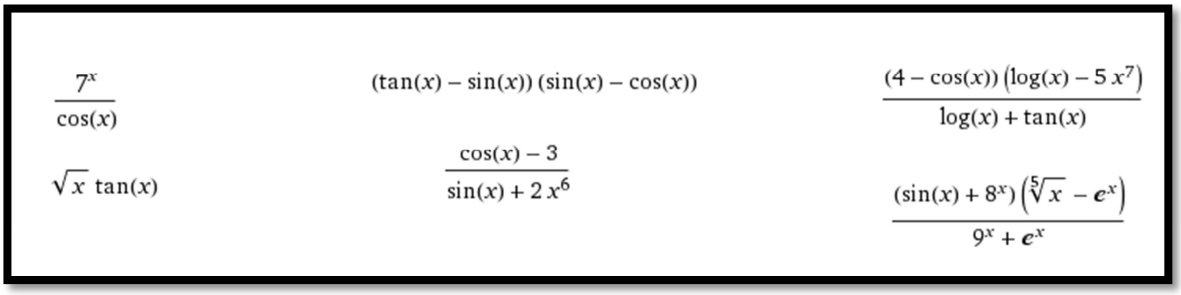

Ilustración 6. Multiplicación y división de funciones

4.1.5 Regla de la cadena

Finalmente, en el momento que el alumnado ya conozca la regla de la cadena para derivar funciones anidadas dentro de otras funciones, se le plantearán ejercicios de este nivel. Como en el nivel anterior, dentro podemos tener varios subniveles. Uno solo con regla de la cadena, inmediatas, sumas y restas. Otro con cualquier tipo de funciones anidadas. Y por último, mezclando también con multiplicación y división. Podemos de nuevo, ver dos ejemplos de cada en la siguiente ilustración:

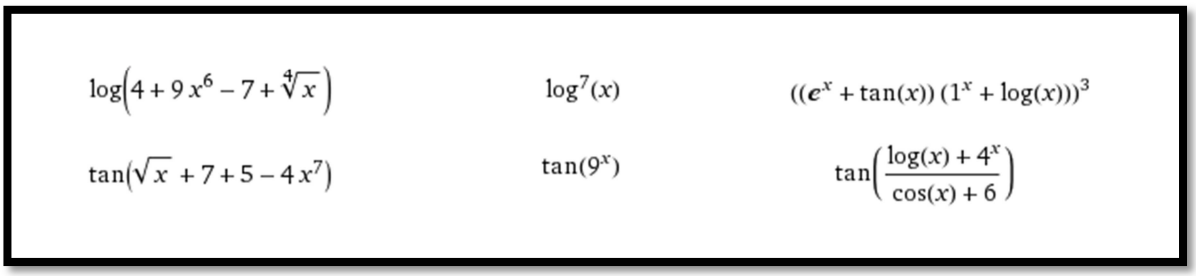

Ilustración 7. Regla de la cadena

4.2 Estructura de la aplicación

Una vez delimitado el ámbito concreto de la aplicación y conocidas las tecnologías con las que se implanta la misma, es el momento de organizar la estructura a utilizar.

Para ello, se hace uso de las tecnologías comentadas anteriormente: JSP, Servlets y JSON. En un primer momento se monta la aplicación de forma que la página inicial (“index.jsp”) te permita seleccionar si quieres acceder con perfil de profesor o de alumno.

A partir de la misma se accede a un Servlet (“LoginServlet.java”), desde el cual, si hemos seleccionado el perfil del profesor, nos envía a un nuevo JSP (“profesor.jsp”), en el que se pueden seleccionar una serie de opciones iniciales para configurar el tipo de ejercicio deseado. Una vez que generamos un ejercicio, pasamos el control a un nuevo Servlet (“GenerarEjercicioServlet.java”), y desde ahí y a través de la librería GSON, se crea el archivo JSON que se usa para almacenar los ejercicios que están disponibles para los alumnos.

A continuación, se le cede el control a “MostrarApartadoServlet.java” que carga en “profesorSol.jsp” el primero de los apartados seleccionados. Si cambiamos de apartado dentro de esta página JSP, volverá a ser el mismo Servlet el encargado de cargar el nuevo apartado. Por último, pulsando en el botón “Inicio” podemos volver a la página de entrada.

Por el contrario, si en la página inicial seleccionamos el perfil de alumno, el Servlet nos manda directamente a una nueva página JSP (“alumno.jsp”), donde estarán cargados los ejercicios almacenados en el archivo JSON comentado anteriormente. Pulsando el botón “Enviar Solución”, accedemos a un Servlet que cargará el primero de los apartados en “alumnoSol.jsp”. Como en el caso anterior podemos cargar otros apartados y volver a la página inicial. Pero además, podemos pulsar el botón “Corregir” que nos enviará al Servlet encargado de corregir: los pasos introducidos y la solución final proporcionada (“CorregirServlet.java”).

En el siguiente esquema se puede observar la estructura de archivos de la aplicación, comentada en las líneas previas:

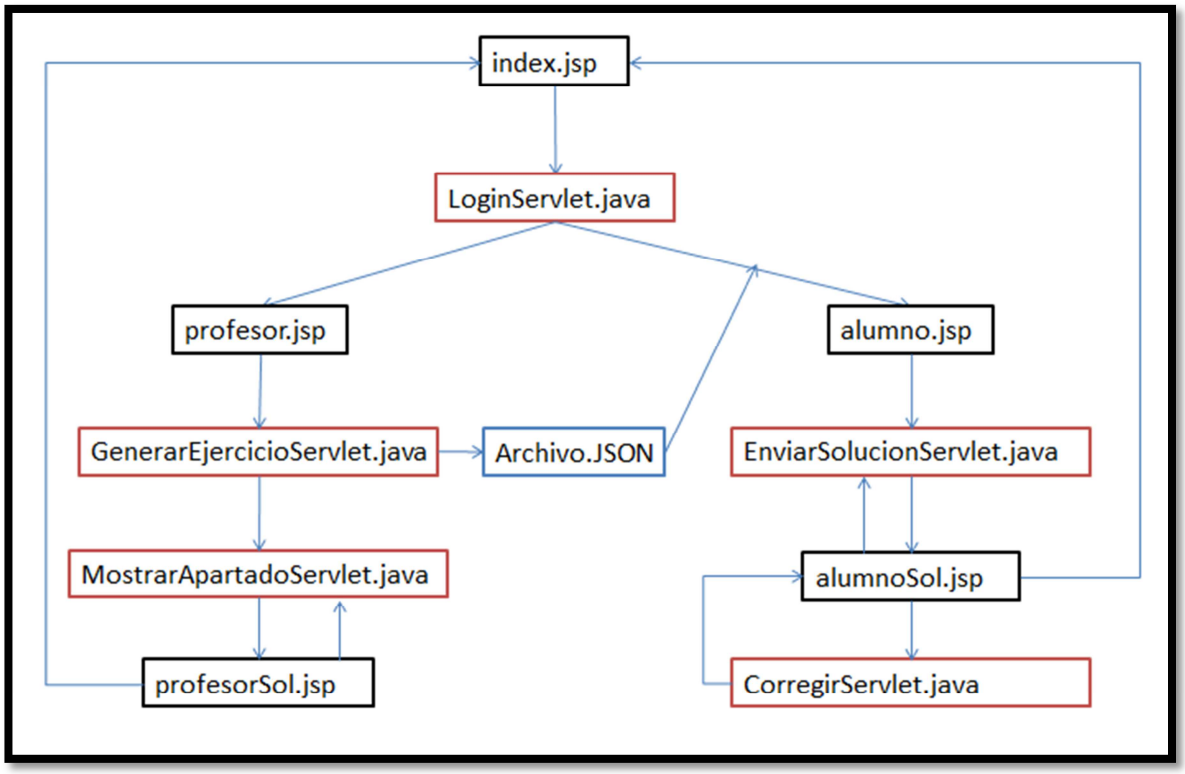

Ilustración 8. Estructura de la aplicación

5 Implementación

En este capítulo se explican los archivos utilizados para la implementación de la aplicación. Estos son: las clases Java, los Servlets y las páginas JSP.

5.1 Clases Java

En este apartado se describirá el funcionamiento de las clases Java usadas en la aplicación. En la siguiente imagen podemos observar las mismas:

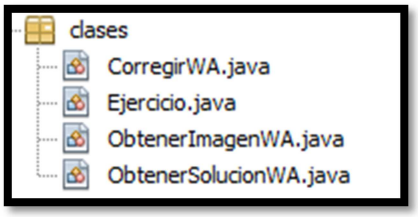

Ilustración 9. Clases de la aplicación

5.1.1 Clase “Ejercicio”

En un primer momento, solo fue necesario crear una clase Java, “Ejercicio”. Era una clase con solo dos atributos, uno de tipo cadena, donde se almacena el enunciado del ejercicio y otro de tipo lista de cadenas donde se guarda cada uno de los apartados del ejercicio.

Se incluyeron también el constructor, que será utilizado al generar los ejercicios, y los métodos “get” para cada atributo, que serán necesarios para recuperar los datos del ejercicio desde la parte de la aplicación correspondiente al alumno.

Como se ha comentado, este intercambio de datos se hace con ayuda de JSON y GSON, con lo cual podemos decir que estamos pasando directamente los objetos de tipo “Ejercicio”. Este aspecto permitió que se pudieran ir mejorando progresivamente las clases necesarias para el funcionamiento de la aplicación.

Y gracias a esto se pudieron incluir varios atributos más. Como fueron dos nuevas listas de cadenas donde almacenar tanto las soluciones de los apartados, como los enlaces a las imágenes de las funciones de forma matemática tradicional. Además de dos listas de listas de cadenas en las que guardar los posibles pasos para dar la solución del apartado y otras formas alternativas de mostrar la solución al mismo. Todo ello con sus métodos correspondientes. La estructura final de esta clase es la siguiente:

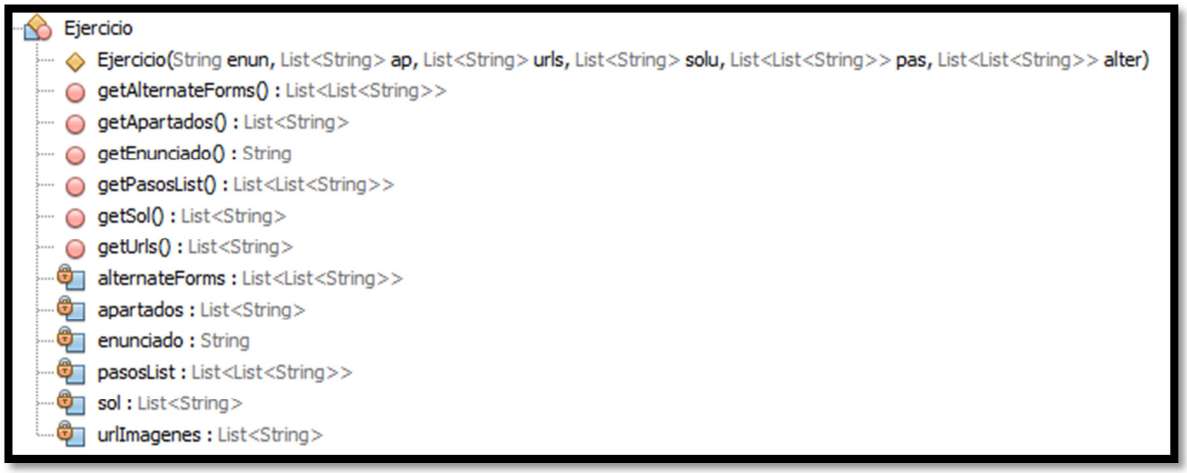

Ilustración 10. Clase Ejercicio

5.1.2 Clases de consultas a Wolfram Alpha

La estructura de las tres clases encargadas de realizar las consultas a Wolfram Alpha son muy similares en alguno de sus aspectos, principalmente a la hora de cómo se realizan dichas consultas. Por otra parte, difieren en la forma en la que se interpretan y tratan los resultados obtenidos.

Para realizar la petición necesitamos los siguientes elementos:

- Texto de la consulta: es la cadena de caracteres acerca de la cual queremos obtener una respuesta de WA, en esta ocasión es pasada como parámetro a cada una de las funciones principales contenidas en las clases.
- APPID: es una ID necesaria para realizar consultas a través de la API de WA. Para su obtención es necesario un registro previo en la web del motor de respuestas.
- Propiedades: son las indicaciones a través de las cuales indicamos el formato en el que queremos las respuestas (texto, imagen, solución por pasos…).

El formato de las respuestas proporcionadas por la API está organizado de la siguiente forma:

- Pods: son los apartados principales en los que se divide la respuesta.
- Subpods: son los subapartados dentro de cada pod.
- Element: son los elementos concretos dentro de cada subapartado.

Dependiendo del formato indicado y del tipo de cadena que pasemos en la consulta, cada uno de estos elementos estará distribuido de una u otra forma.

En los siguientes capítulos veremos cómo se realizan estas consultas para cada una de las clases creadas en la aplicación, así como la interpretación de las respuestas.

5.1.3 Clase “ObtenerSolucionWA”

En esta ocasión el texto de la consulta es la propia función a derivar, a la que se añaden sendos paréntesis a izquierda y derecha, más una comilla simple para indicar que queremos la derivada de esa función.

Además, en las propiedades de la consulta indicaremos que sólo queremos respuestas en modo texto y que necesitamos la solución paso a paso de la derivada en los casos que sea posible.

Para interpretar la respuesta la exploraremos hasta llegar al pod titulado “Derivative”, y una vez dentro obtendremos de los subpods correspondientes, tanto la solución como los posibles pasos intermedios. Para conseguir las formas alternativas de la solución deberemos buscar el pod llamado “Alternate forms”.

Para finalizar la descripción de esta clase hay que indicar que se incluyen dos métodos para convertir las cadenas de la respuesta a un formato más sencillo de tratar y de mostrar en la aplicación. Además de los métodos “get” correspondientes para obtener cada uno de los atributos que guardamos en esta clase. Su estructura final queda como se ve en la siguiente imagen:

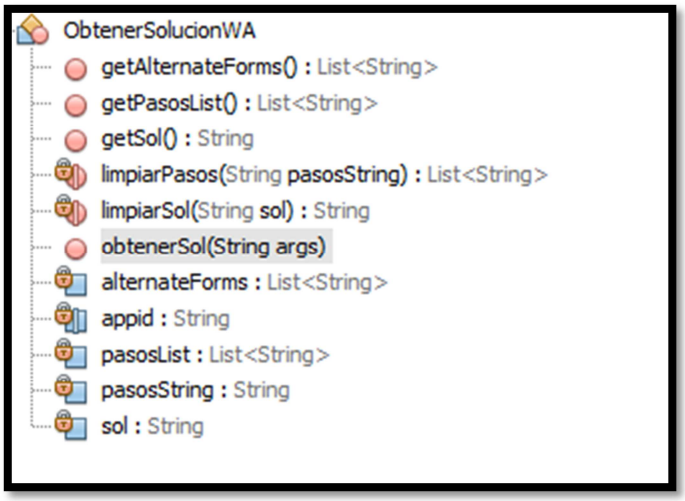

Ilustración 11. Clase "ObtenerSolucionWA"

5.1.4 Clase “ObtenerImagenWA”

El propósito de esta clase es proporcionar un método que pasándole como parámetro una función, nos devuelva la url a la imagen de la función generada por WA.

Por tanto, el texto de la consulta será la propia función de la que queremos obtener su imagen. Y en las propiedades de la misma indicaremos que queremos la respuesta tanto en texto como en imagen.

Para obtener la parte de la respuesta que nos interesa, debemos buscar el pod “Input” y dentro de sus subpods, obtener el elemento de tipo imagen, al que a través del método correspondiente (“getURL()”), podemos obtener el enlace a dicha imagen.

La estructura de esta clase es mucho más sencilla que la anterior:

Ilustración 12. Clase "ObtenerImagenWA"

5.1.5 Clase “CorregirWA”

Esta clase contiene un método principal que recibe como parámetros dos funciones. La cadena principal de la consulta será la evaluación de si ambas funciones son equivalentes. En las propiedades de la consulta indicaremos que queremos el resultado en modo texto únicamente. Y el propósito de la función será buscar si el estado del pod “Result” es “True”. Al igual que la clase anterior su estructura es bastante simple:

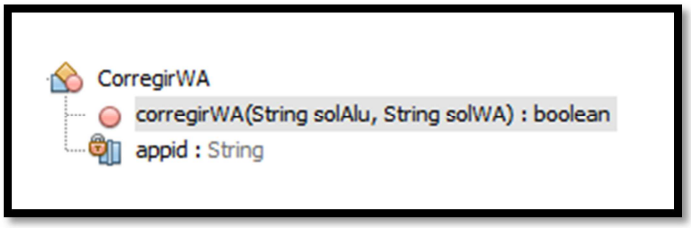

Ilustración 13. Clase "CorregirWA"

5.2 Servlets

El propósito de este apartado es proporcionar una idea del funcionamiento de los Servlets utilizados en la aplicación, haciéndolo con una extensión más amplia en aquellos que cobran más importancia a la hora de realizar las tareas principales de la misma.

Ilustración 14. Servlets

5.2.1 Generar Ejercicio Servlet

Una vez que teníamos montada la estructura de la aplicación y que podíamos pasar objetos de la clase “Ejercicio” a través del archivo JSON, llegó el momento de generar esos ejercicios de forma automática. Antes de entrar en detalle sobre cómo se ejecuta esa tarea, es necesario describir las opciones que tiene el profesor para configurar la forma en la que serán generados los ejercicios.

Lo primero que se ha de configurar es el enunciado del ejercicio y el número de apartados del mismo. Además, se disponen de tres listas desplegables que son las que determinarán de qué forma serán las funciones que se han de generar. Estas son:

- Tipo (Cualquiera; Constantes, polinómicas y raíces; Exponenciales logarítmicas y trigonométricas): Es una primera forma de definir si queremos funciones de cualquier tipo, o de alguno de los conjuntos descritos en las opciones.
- Dificultad (Inmediatas; Sumas y restas; Multiplicación y división; Regla de la cadena): son las dificultades progresivas que se han explicado anteriormente en el apartado de aproximación matemática.
- Nivel (Fácil; Medio; Difícil): Dentro de una de las dificultades anteriores, se puede escoger que las funciones sean de mayor o menor nivel.

Estas opciones están incluidas en la página “profesor.jsp”, y son pasadas como parámetros a “GenerarEjercicioServlet.java”. Para generar las funciones, el Servlet realiza las siguientes tareas:

- En primer lugar, hay que crear una lista tipo “String”, donde ir almacenando cada uno de los apartados correspondientes.
- Posteriormente, habrá que crear tantos apartados como se haya seleccionado anteriormente. Dependiendo de la dificultad escogida se llamará a un método que generará el tipo de función escogida.
- Método “seleccionaFuncion”: Es un método que devuelve una función cuya derivada es inmediata, dependiendo del tipo escogido podrá devolver una de las siguientes: constante, monomio, raíz, un número elevado a x, e elevado a x, logaritmo neperiano, seno, coseno o tangente.
- Método “seleccionaFuncionSumRes”: En este caso devuelve una suma o resta de las funciones que genera el método anterior. Dependiendo del nivel escogido la longitud de esta función será mayor o menor.
- Método “seleccionaFuncionSumRes”: Método que devuelve funciones con multiplicaciones y divisiones. Dependiendo del nivel escogido devolverá:
    - Fácil: Multiplicación o división de dos funciones simples.
    - Medio: Multiplicación o división de dos sumas o restas de funciones simples.
    - Difícil: Multiplicación y división de sumas y restas de funciones simples.
- Método “seleccionaFuncionCompuesta”: devuelve una función en la que tendremos que aplicar la regla de la cadena para realizar su derivada. En este caso el nivel influye de la siguiente forma:
    - Fácil: Genera una suma de constantes, monomios o raíces, que a su vez estarán dentro de un exponente, logaritmo o función trigonométrica.

- Medio: Devuelve dos funciones anidadas cualquiera de las generadas por “seleccionaFuncion”.
- Difícil: Genera una multiplicación o división de funciones, anidadas dentro de un exponente, logaritmo o función trigonométrica.

Además de todo lo explicado con anterioridad, para cada apartado será necesario el uso de las clases “ObtenerSolucionWA” y “ObtenerImagenWA” ya explicadas anteriormente.

Finalmente, una vez que ha sido generado completamente el ejercicio, se almacena en el archivo JSON con la ayuda de la librería GSON, y se cede el control a “MostrarApartadoServlet”.

5.2.2 Corregir Servlet

Este es el Servlet encargado de recoger la solución propuesta por el alumno y decidir si es correcta o no. Para ello, cogerá tanto los pasos introducidos por el alumno como la solución final, y los irá corrigiendo haciendo uso de la clase “CorregirWA”.

En primer lugar, es necesario recoger todos los atributos y parámetros correspondientes para realizar la tarea. A continuación, se comparan los pasos propuestos por el alumno en caso de que los haya introducido. Para esta tarea cada paso se compara con el anterior, con el fin de que el alumno pueda saber en cuál de ellos ha cometido el error. Finalmente, se comprueba que la solución propuesta sea solución al enunciado original del apartado.

Después de realizar esta tarea de corrección, se vuelven a enviar todas las variables utilizadas, en forma de atributo, para que las recoja la página JSP correspondiente, en este caso “alumnoSol.jsp”.

5.2.3 Login Servlet

Es el Servlet encargado de realizar las tareas necesarias una vez que se selecciona un tipo de usuario en la página de inicio.

En caso de que la opción escogida sea la de “Profesor” simplemente tendrá que redirigir a la página “profesor.jsp”.

Por el contrario, si la opción seleccionada es la de “Alumno”, deberá cargar el ejercicio contenido en el archivo JSON, para después enviarlo como atributo a la página “alumno.jsp”.

5.2.4 Mostrar Apartado Servlet

Nos encontramos ahora con un Servlet encargado de realizar dos tareas.

La primera de ellas, consiste en cargar el primero de los apartados justo después de que se genere el ejercicio. Es decir, una vez que el profesor ha seleccionado los parámetros para la generación del ejercicio, y esta tarea ya ha sido realizada, este Servlet toma el control de la aplicación y carga el primero de esos apartados en “profesorSol.jsp”. Esta tarea se realiza a través del método “GET”.

Por otro lado, si este Servlet es llamado a través de su método “POST”, quiere decir que ha sido la página “profesorSol.jsp” la que requiere su uso. En este caso, la labor será la

de cargar el apartado concreto que le sea requerido a través del atributo correspondiente y volver a enviarlo a la página anteriormente nombrada.

5.2.5 Enviar Solución Servlet

Éste sería el Servlet correspondiente al anterior en la parte de la aplicación dedicada al alumno. La primera vez que se llama cargará el primero de los apartados en "alumnoSol.jsp"; el resto de veces será esta propia página la que lo llamará a través de su método "POST", para cargar el apartado deseado.

5.3 Páginas JSP

En este capítulo se hará una breve descripción de cada una de las páginas JSP incluidas en el trabajo, que se acompañarán de una captura de cada una de ellas antes de aplicarle ningún tipo de estilo. Son las siguientes:

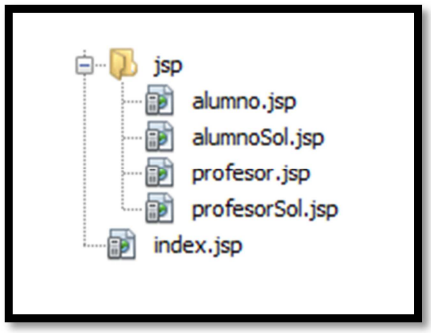

Ilustración 15. Páginas JSP

En primer lugar tenemos "index.jsp", que se encarga simplemente de proporcionar una interfaz que permita seleccionar si el usuario actual es un profesor o un alumno. Es una tarea que en un uso real de la aplicación debería ser llevada a cabo por la plataforma educativa en la que esté ubicada. Posteriormente, pasa el control de la aplicación a "LoginServlet.java":

Ilustración 16. Index.jsp

Pasemos a continuación a ver las páginas JSP correspondientes a la parte de la aplicación usada por un docente. Tendríamos primero “profesor.jsp”, que se encarga de mostrar las opciones de configuración del ejercicio a generar explicadas anteriormente:

Ilustración 17. Profesor.jsp

También disponemos de la página “profesorSol.jsp”, que como hemos comentado con anterioridad, muestra los apartados ya creados, junto con su solución, y los posibles pasos para alcanzarla, así como otras formas alternativas en caso que estén disponibles:

Ilustración 18. ProfesorSol.jsp

Con respecto a las JSP que se usan en la parte de la aplicación utilizada por el alumno, tenemos en primer lugar “alumno.jsp” que nos muestra todos los apartados del ejercicio que haya actualmente creado:

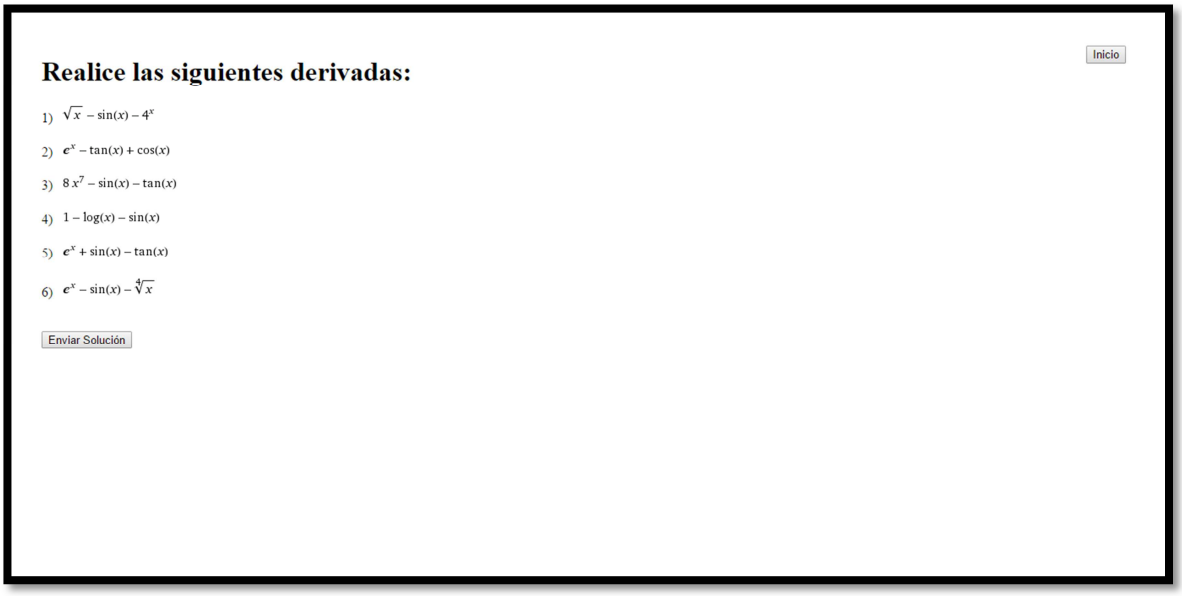

Ilustración 19. Alumno.jsp

Por otro lado, tenemos “alumnoSol.jsp” que ya nos proporciona la interfaz necesaria para que el alumno aporte su posible solución al ejercicio:

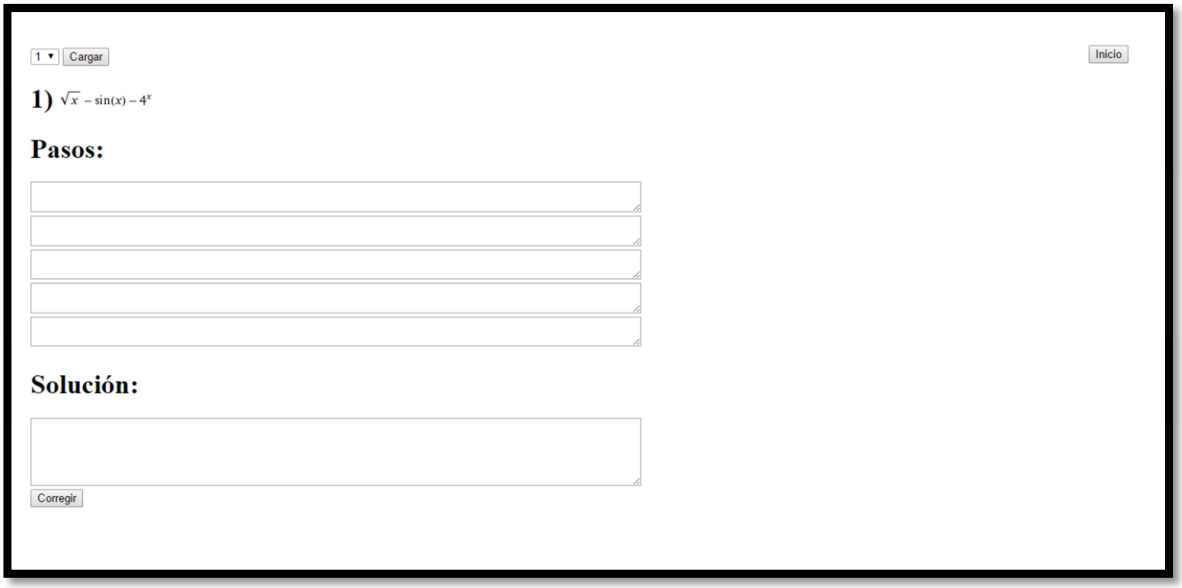

Ilustración 20. AlumnoSol.jsp

5.4 Diseño gráfico de la aplicación

Como se ha comentado anteriormente, el diseño gráfico de la aplicación es la última fase desarrollada en este trabajo. Esta tarea ha sido llevada a cabo usando las tecnologías CSS y Bootstrap.

Los objetivos al desarrollar esta fase son: dotar a la aplicación de una interfaz más amigable, que siga la misma línea en todas sus páginas y que contenga un tipo de diseño “Responsive”.

Para ello se han realizado las siguientes labores:

- Agregar en la cabecera de cada página JSP la siguiente etiqueta:

```
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
```

- Introducir en cada página una cabecera con las siguientes características:
    - Pre-título con el nombre de la aplicación.
    - Título de la página JSP en la que nos encontramos.
    - Botones de navegación, en caso que sea necesario.
- Con respecto a los estilos de cada elemento se han ejecutado estas dos tareas:
    - Introducir en cada elemento HTML la clase Bootstrap que se quisiese representar, a través de su atributo “class”.
    - Para el resto de modificaciones estéticas, se ha usado el atributo “style”, introduciendo en el mismo el código CSS correspondiente.
- Finalmente, para la distribución de los elementos en la pantalla se ha utilizado la etiqueta “ ”, agregando también el atributo “class” en caso de que se quisiese seguir alguna de las clases Bootstrap disponibles.

6 Conclusiones

Uno de los últimos puntos de esta memoria debe hacer referencia a las conclusiones y a las posibles mejoras que se podrían realizar sobre la aplicación final, así como a las dificultades encontradas durante el desarrollo de la misma.

Este trabajo surgió con el propósito de poder proporcionar una herramienta de enseñanza y aprendizaje, tanto a docentes como al alumnado en niveles de Bachillerato, de alguno de los contenidos impartidos en dichos cursos. Es ahora el momento de valorar si esos objetivos se han conseguido.

En primer lugar se van a describir las dificultades encontradas, posteriormente se procederá a presentar un conjunto de conclusiones y en base a ellas, se detallarán una serie de posibles mejoras.

6.1 Dificultades encontradas

Las principales dificultades a la hora de implementar la aplicación las podemos agrupar dentro del estudio y aplicación de las dos herramientas que han sido necesarias estudiar desde cero para la realización de este trabajo: JSON y la API de Wolfram Alpha.

Con respecto a JSON, podemos decir que ha resultado ser una maravillosa herramienta para el paso de datos entre dos aplicaciones, junto a la librería GSON que proporciona los métodos necesarios para llevar a cabo esa tarea de forma mucho más sencilla.

En lo que respecta a la comunicación con WA, la tarea ha sido un poco más tediosa. En primer lugar fue necesario familiarizarse con el tipo de respuestas que proporciona dicha API, labor facilitada por el explorador interactivo que podemos encontrar en la página del motor de respuestas (referencia bibliográfica número 3).

Por otro lado, también hubo que hacer un diseño adecuado de las consultas, además de las clases Java para ejecutar dichas consultas, faena facilitada por las librerías proporcionadas en la web de WA (referencia bibliográfica número 4).

6.2 Conclusiones

Las conclusiones las abordaremos desde dos puntos de vista. El primero, la comprobación de que los objetivos iniciales se han cumplido. El segundo, la valoración personal de las aptitudes adquiridas durante la elaboración del trabajo.

Con respecto a los puntos que se marcaron inicialmente que cubriese la aplicación, podemos afirmar que se han cumplido todos ellos, y son:

- Aplicación que ayude a la enseñanza y aprendizaje de alguno de los contenidos impartidos en la asignatura de matemáticas de nivel de Bachillerato.
- Generación automática de ejercicios en base a ciertos parámetros.
- Interfaz para poder ver las soluciones y los posibles pasos para la solución de los ejercicios.
- Corrección automática de soluciones paso a paso.
- Interfaz para proporcionar soluciones y comprobar las correcciones.

En lo que respecta a la valoración personal de la elaboración del trabajo, puedo decir que me ha ayudado a seguir completando mi formación tras el estudio de todas las

asignaturas gracias a todos los conocimientos adquiridos durante la elaboración de este trabajo. También me ha servido para darle una aplicación práctica a lo aprendido durante esas asignaturas. Y finalmente, también ha sido una buena experiencia a la hora de trabajar de forma autónoma.

6.3 Líneas Futuras

Con respecto a los puntos en los que se podría seguir trabajando en la línea de este trabajo, surgen varias ideas:

- En primer lugar, hacer una adaptación definitiva de esta aplicación para su uso dentro de una plataforma educativa, para poder aprovechar todo el rendimiento de la misma.
- Por otro lado, también se podría trabajar en la inclusión de otros contenidos pertenecientes a un nivel similar de estudios. Por ejemplo:
    - Límites.
    - Operaciones con matrices.
    - Integrales.
- Pequeñas mejoras dentro de la aplicación:
    - Incluir una guía sobre cómo introducir las soluciones, o cómo expresar los pasos.
    - Proporcionar un teclado en pantalla para facilitar la introducción de las soluciones.
    - Aumentar el número de parámetros que se pueden seleccionar para generar los ejercicios.

7 Referencias bibliográficas

1. Página principal de Wolfram Alpha:
   https://www.wolframalpha.com/
2. Documentación de la API de Wolfram Alpha:
   http://products.wolframalpha.com/api/documentation.html
3. Explorador interactivo de la API de Wolfram Alpha:
   http://products.wolframalpha.com/api/explorer.html
4. Librerías de la API de Wolfram Alpha:
   http://products.wolframalpha.com/api/libraries.html
5. Página principal de JSON:
   http://json.org/
6. Página principal de GSON:
   https://sites.google.com/site/gson/Home
7. Información sobre Servlets:
   https://jtagua.wordpress.com/2010/10/31/tutorial-de-servlet-1-introduccion-ciclo-de-vida-y-ejemplo-basico/
8. Página principal de JSP:
   http://www.oracle.com/technetwork/java/javaee/jsp/index.html
9. Página principal de Apache Tomcat:
   http://tomcat.apache.org/
10. Página con multitud de tutoriales en programación web:
    http://www.w3schools.com/

8 Anexos técnicos

El capítulo final de esta memoria se reserva para los anexos técnicos, en este caso se trata del manual de la aplicación desarrollada.

8.1 Manual de la aplicación

A continuación se describe con detalle la funcionalidad y modo de uso de la aplicación que se ha desarrollado.

8.1.1 Pantalla de inicio

Es la primera pantalla que nos encontramos al abrir la aplicación. En ella tenemos dos opciones, que nos llevarán a una u otra pantalla tras pulsar el botón “Aceptar”:

- Profesor: Nos lleva a la parte de la aplicación gestionada por un docente, concretamente a la página necesaria para crear un nuevo ejercicio.
- Alumno: Nos lleva a la parte de la aplicación gestionada por un alumno, concretamente a la página necesaria para ver el ejercicio que haya propuesto en ese momento.

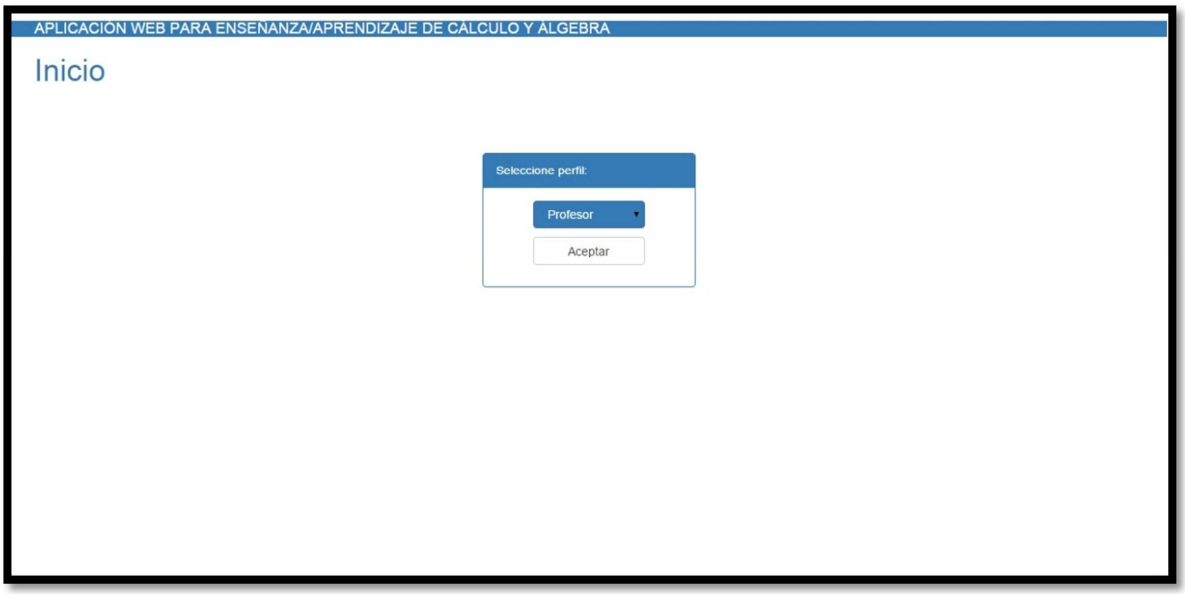

Ilustración 21. Pantalla de inicio

8.1.2 Profesor – Generar Ejercicio

En esta página nos encontramos con un formulario que nos permite generar un nuevo ejercicio a partir de ciertos parámetros:

- Enunciado. Se trata del enunciado del ejercicio a generar.
- Número de apartados. Podemos definir el número de apartados que deseamos que la aplicación genere.
- Tipo de funciones. Tenemos tres opciones a escoger:
  - Cualquiera.
  - Constantes, polinómicas y raíces.
  - Exponenciales, logarítmicas y trigonométricas.

- Dificultad. Podemos escoger entre:
    - Inmediatas.
    - Sumas y restas.
    - Multiplicación y división.
    - Regla de la cadena.
- Nivel. Dentro de las dificultades anteriores podemos escoger nivel fácil, medio o difícil.

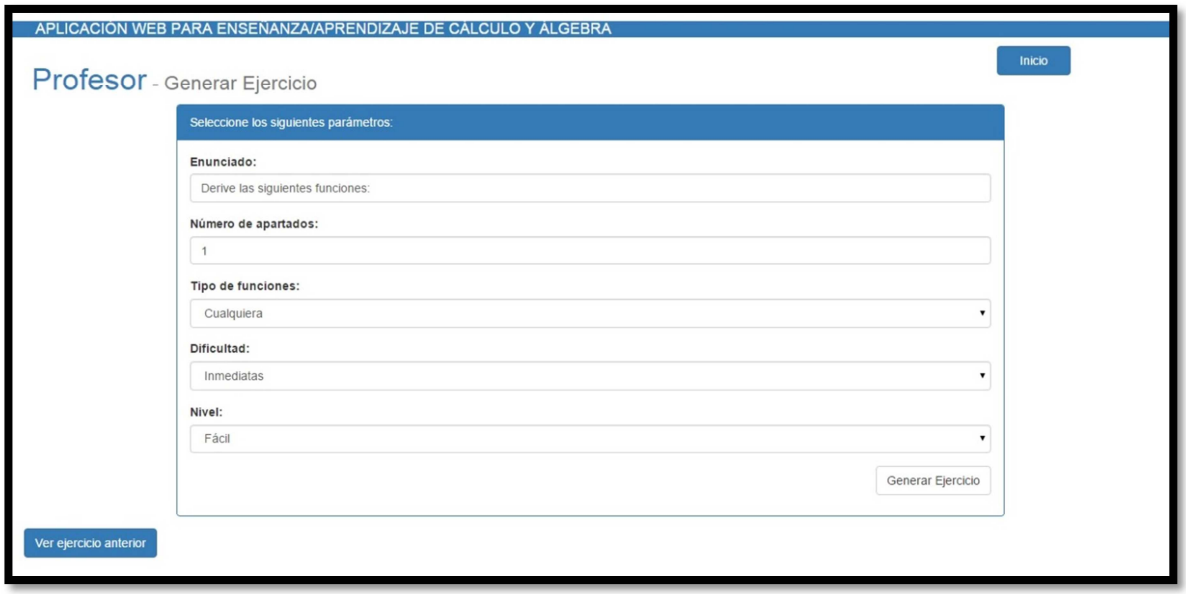

Ilustración 22. Profesor - Generar Ejercicio

8.1.3 Profesor – Ver apartado

A esta página podemos acceder por dos motivos. El primero de ellos, porque se acabe de generar un ejercicio. El segundo, porque se haya pulsado el botón “Ver ejercicio anterior”. En ambos casos, la página carga el primero de los apartados del ejercicio en cuestión. En caso que el ejercicio tenga más de un apartado podemos cargarlo seleccionando su número en la lista desplegable correspondiente, y pulsando el botón cargar.

En esta página siempre tendremos los dos siguientes campos:

- Apartado. Se trata de la función generada para su posterior derivación.
- Solución. Se trata de la función anterior ya derivada, una solución propuesta por WA.

En caso que estén disponibles puede haber uno, otro o ambos de los siguientes:

- Pasos. Se trata de una secuencia de posibles pasos para alcanzar la solución, propuestos por WA.
- Formas Alternativas. Son distintas formas matemáticas de expresar la solución.

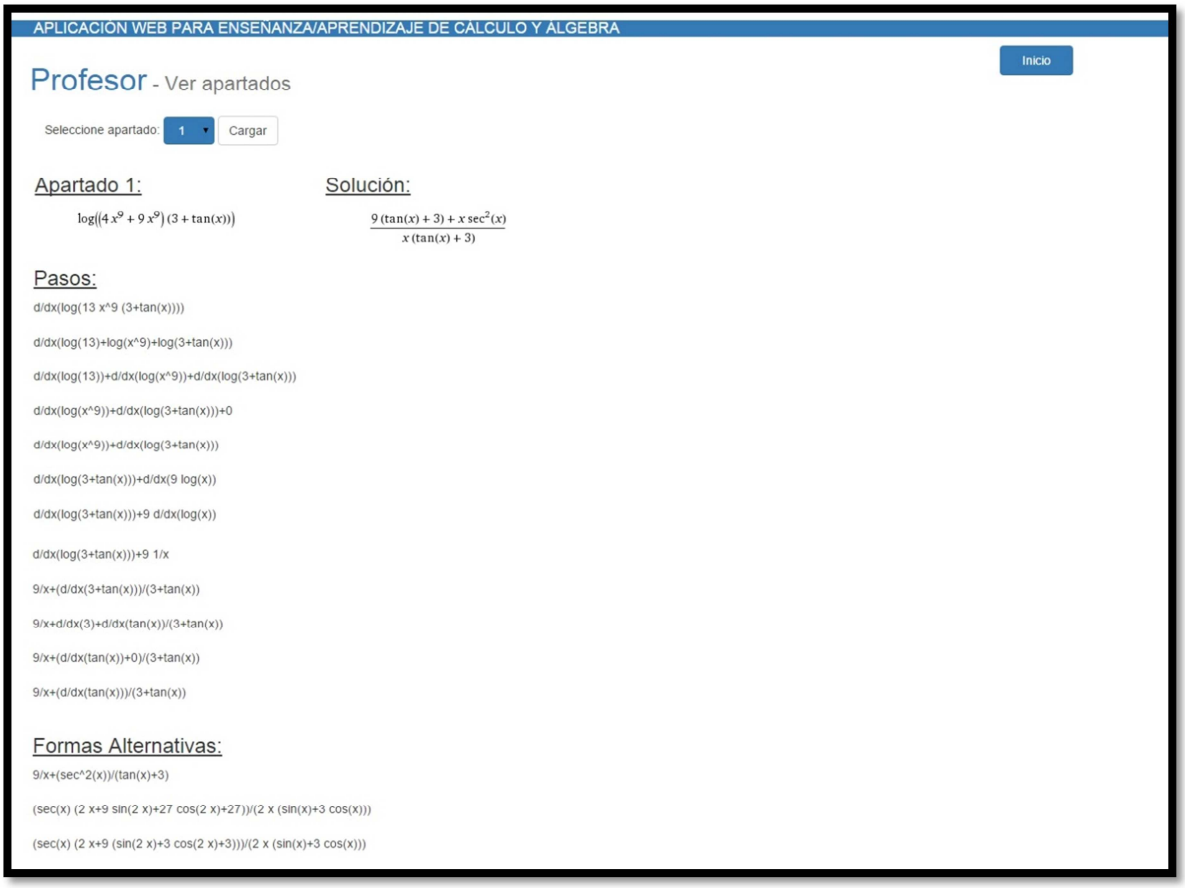

Ilustración 23. Profesor - Ver apartado

8.1.4 Alumno – Ver ejercicio

A esta página se accede cuando escogemos la opción “Alumno” en la pantalla inicial. En la misma se muestran todos los apartados del ejercicio que haya propuesto en ese momento.

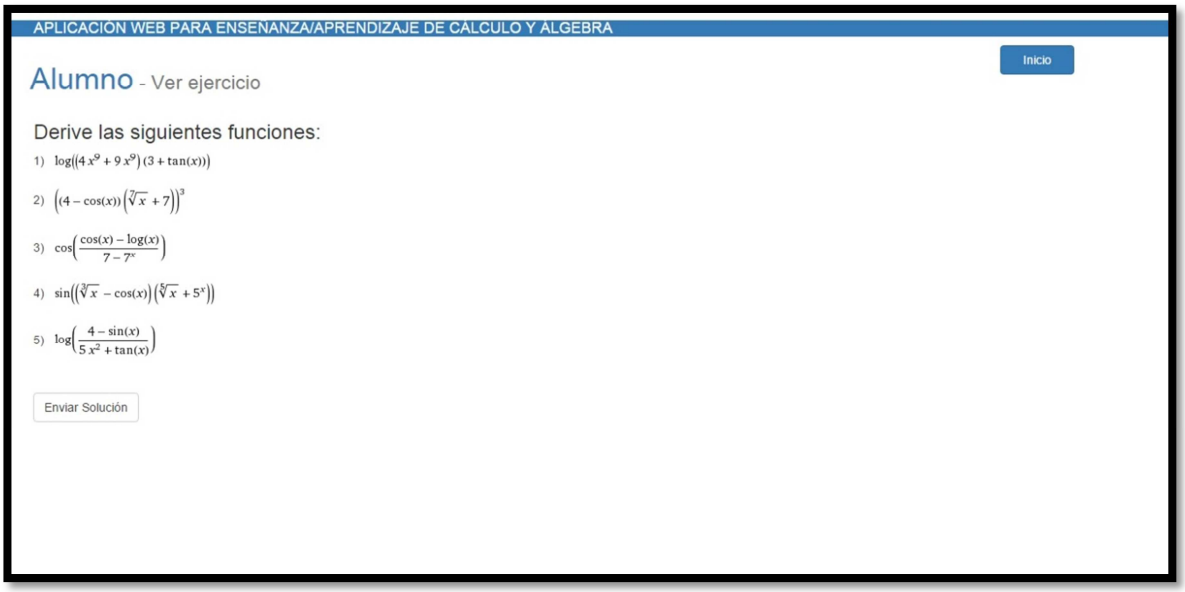

Ilustración 24. Alumno - Ver ejercicio

8.1.5 Alumno – Enviar solución

En esta página se cargará el primer apartado disponible del ejercicio propuesto. Podemos cargar otros apartados seleccionando su número en la lista desplegable y pulsando el botón “Cargar”.

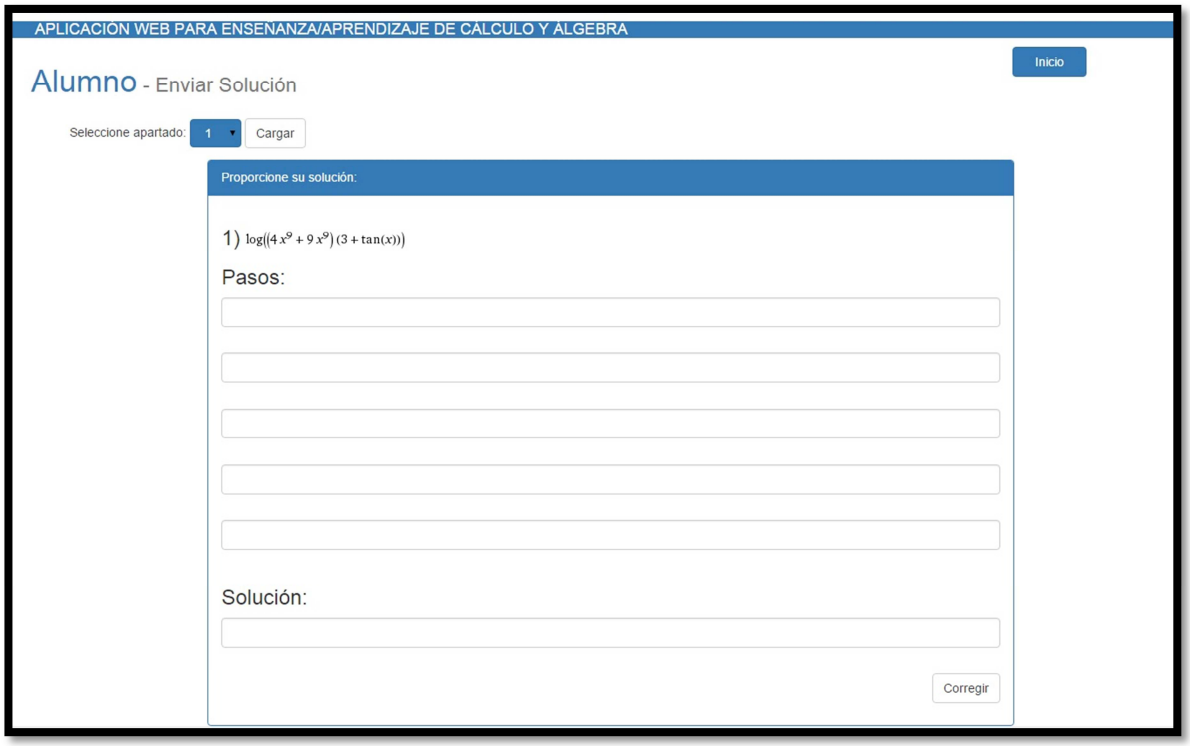

Ilustración 25. Alumno - Enviar solución

Para enviar una solución debemos opcionalmente rellenar los campos marcados con la palabra “Pasos”. Y obligatoriamente el campo marcado como “Solución”. Al pulsar el botón “Corregir”, la página se actualizará indicando si cada paso es correcto con respecto a su anterior y si la solución propuesta, en realidad resuelve el enunciado original del apartado. Podemos observar un ejemplo en la ilustración 26.

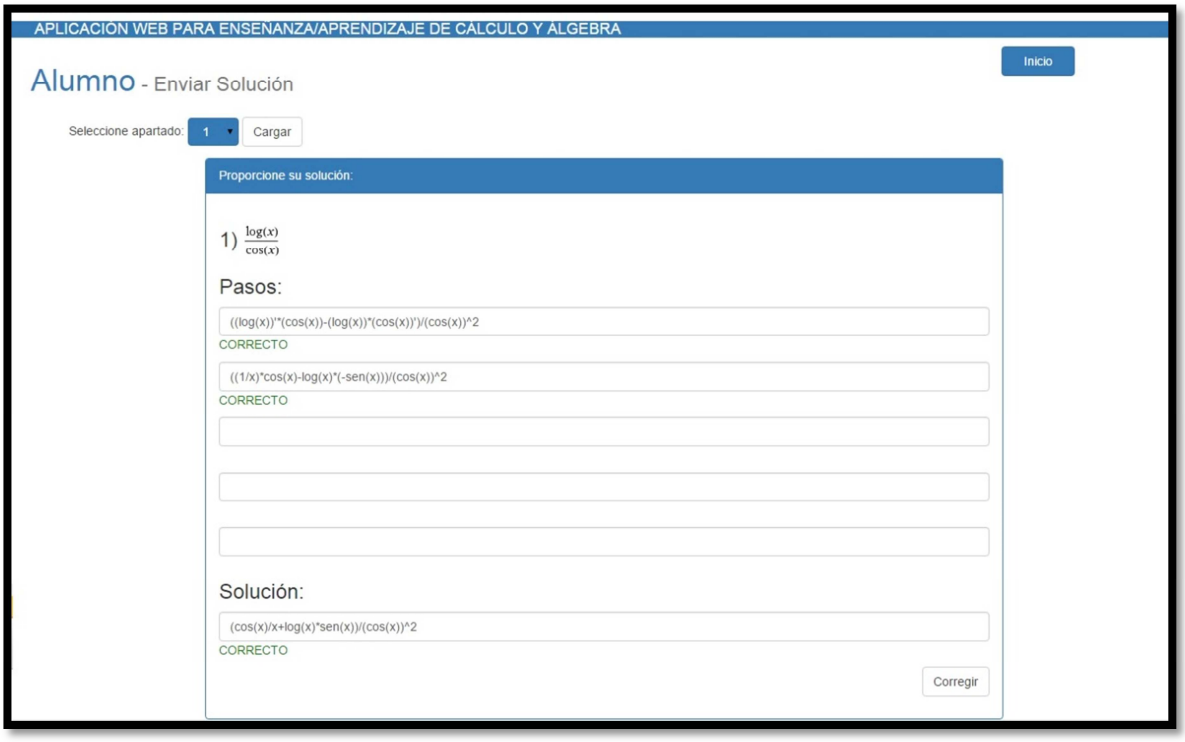

Ilustración 26. Alumno - Corrección